Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 245 – 252

Procedia Computer Science

The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

Simulation-based Verification of Automotive Safety-Critical Systems based on EAST-ADL

Ralph Weissnegger[a,b,*], Markus Schuss[a], Christian Kreiner[a], Markus Pistauer[b], Kay Römer[a], Christian Steger[a]

[a] *Institute for Technical Informatics, Graz University of Technology (TU Graz), Austria*
[b] *CISC Semiconductor GmbH, Klagenfurt, Austria*

Abstract

The increasing amount of assistance features in today's vehicles to ensure safe and reliable operation, imply increasingly complex systems. New challenges are arising due to highly heterogeneous and distributed systems which interact with and have an impact on the physical world, so called cyber-physical systems. Since millions of test kilometers must be driven to ensure a reliable system, simulation-based verification is becoming more important to reduce costs and time-to-market. This situation prompts the urgent demand for new techniques to simulate the behavior in early development phases by reusing verified system components. Best combined within a model-based approach that both unites different stakeholders and helps non-specialists to understand problems in the design. In this paper, we present a novel method for simulation-based verification of automotive UML/EAST-ADL design models. To demonstrate its benefits, our methodology is applied in an industrial use case of a battery management system.

Peer-review under responsibility of the Conference Program Chairs
Keywords: UML; EAST-ADL; automotive; ISO26262; simulation; verification; SystemC;

1. Introduction

Today's cars consist of highly complex electric/electronic (E/E) systems with sensors and actuators networking with each other. In fact a car is now more or less a smartphone on wheels. It can be observed that there is a shift towards fully E/E cars, since electric cars are getting more popular. The sensing and controlling of these systems is the work of the highly distributed electrical control units (ECU) and it is no surprise that more than 200 of these micro-controllers are currently integrated in a modern electric vehicle[1]. Since the electrification in the automotive domain continuous, new challenges in the development process are arising. This is especially the case where multiple stakeholders including specialists for hardware, software and system design have to work together with safety engineers to ensure a reliable and safe system. A model-based approach helps non safety-specialist to also understand problems in the design and development of safety-critical systems. One modeling languages which has established

* Corresponding author: Ralph Weissnegger
E-mail address: ralph.weissnegger@tugraz.at

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2016.04.122

itself in the automotive domain is EAST-ADL[2]. It allows the detailed design of automotive E/E systems on different levels of abstraction. The last two layers conform with the AUTOSAR standard. Furthermore, they are in line with the development process of safety-critical systems according to ISO26262[3] and allow the evaluation and formal verification of design models. Since millions of test kilometer must be driven to ensure a reliable system, simulation is becoming more and more important[4], because it is no longer possible to cover the costs of physical tests. A drawback in today's development process is that simulation tools are often detached from the design tools and require cumbersome imports and exports of files between the environments. It is important that simulation tools are tightly and seamlessly integrated into the design and development process[5], meeting the requirements of ISO26262. Furthermore, approaches are needed that allow a high traceability to requirements and make it possible to derive requirements from simulation-results. This must be done as early as possible and on different abstraction levels.

In this work, we present a model-based simulation framework for the verification of E/E systems in the automotive domain. We link quickly executable simulation models, implemented in SystemC (-TLM) and SystemC-AMS, with EAST-ADL design models. The level of granularity of the models can be easily switched depending on the complexity. Using these reusable components, we achieve an early behavior simulation of the whole system. The result is a tool-aided methodology built as an Eclipse plugin in Papyrus[6], which makes it easy to verify the behavior of automotive safety-critical systems.

2. Related Work

An approach for generating simulation models from EAST-ADL architecture models was presented in[7]. In this work, several architecture levels of EAST-ADL have been mapped to abstraction levels of SystemC-TLM. The architecture of an automotive use case was presented on analysis and design level. For the expression of the behavior, the authors used SystemC code and state machines. This approach works very well for the digital domain, but lacks proper definition needed for analog and mixed-signal components. Through the use of code generators, it is possible to achieve synthesis of very detailed EAST-ADL models. It would also benefit of analyze and verification mechanisms for their simulations.

The authors of[8] presented three different analysis techniques for architectural models described in EAST-ADL, to guarantee the quality in the context of ISO26262. One of the proposed techniques is the simulation of EAST-ADL functions in Simulink. The behavior of each function was linked to FMU or Simulink models to facilitate the simulation. The authors also described mapping rules for the EAST-ADL to Simulink transformation (one-to-one mapping). The results of the simulation have been traced back to the requirements. This approach was applied to an industrial use case of a brake-by-wire system on Design Level. However, in contrast to our approach, they use proprietary simulation engines with high license costs and external tools which are not integrated into the design and development flow.

The authors of[9] demonstrated how to use MARTE for hardware design and simulation. They introduced a step-by-step methodology for hardware modeling with Hardware Resource Models (HRM) stereotypes. The platform models are refined until the final platform class is reached. In a later step, these models are used to generate code with the help of a Java plugin. A tool called Simics was used to facilitate the simulation. Instead of using the whole MARTE spectrum for simulation, this approach only uses HRM models for code generation of very detailed platforms instead of system level design.

3. Model-based System Design

Model-based design plays an ever increasing role in today's development to deal with complex systems. Organization of specialized people in projects of a certain size requires a lot of effort. Therefore, it is becoming increasingly important that stakeholders from different domains, e.g. hardware, software, safety or even security can efficiently work together. Particularly in the evaluation of safety-critical systems, safety specialists need a entire view of the system, that includes all domains of the system. Best combined in a tool where even entire processes like the ISO26262 can be addressed.

One modeling-language which has established itself in the automotive domain is EAST-ADL. It allows the capturing of detailed automotive electric and electronic systems on five layers of abstraction, each with a clear separation

of concerns: *Vehicle* , *Analysis*, *Design* , *Implementation* and *Operational Level*. Besides structural aspects, this modeling language allows the expression of behavior, requirements, verification and validation. The highest level is the *Vehicle Level*, that describes electronic features to allow integration of product variability. The *Analysis Level* includes the Functional Analysis Architecture (FAA), which allows an abstract functional representation of the architecture (what the system shall do), in relation with the features from the *Vehicle Level*. The *Design Level* allows the decomposition of models in the FAA to Functional Design Architecture (FDA) models and Hardware Design Architecture (HDA) models. Within these models the functional representation of the architecture can be allocated onto the hardware platforms. The applications are represented by *DesignFunctionTypes* with annotated behavior and configurations. The hardware components are modeled by *Sensors*, *Nodes* (ECUs), *Actuators* and *HardwarePortConnector* (Buses) and more. They are interconnected by *IOHardwarePins* or *CommunicationHardwarePin* and wired by *HardwareConnectors*. The last two layers (Implementation, Operational) are the realization of the implementation in AUTOSAR. Therefore, the models on these levels are compliant with the AUTOSAR specifications. The behavior of components on all these levels are not explicitly addressed in EAST-ADL. It can be either expressed by behavioral diagrams (state machines, activity diagram) or externally in tools like Matlab.

EAST-ADL as automotive modeling language also addresses parts of the functional safety standard ISO2626, which was one of the outcomes of international projects like ATESST[10] and MEANAD[11]. This enables the language for safety-analysis like Fault Tree (FTA) or Failure Mode and Effect Analysis (FMEA), but also for defining safety-requirements and to achieving high traceability to models and behavioral diagrams. Furthermore this language provides means to describe validation and verification activities by *VVCases*. Since EAST-ADL is included in an Eclipse UML2 Editor called Papyrus[6], it makes it easy to design complex systems without licensing costs.

In the next section, we present the simulation core and how to execute EAST-ADL models with behavioral languages such as SystemC and SystemC-AMS.

4. Executable Models

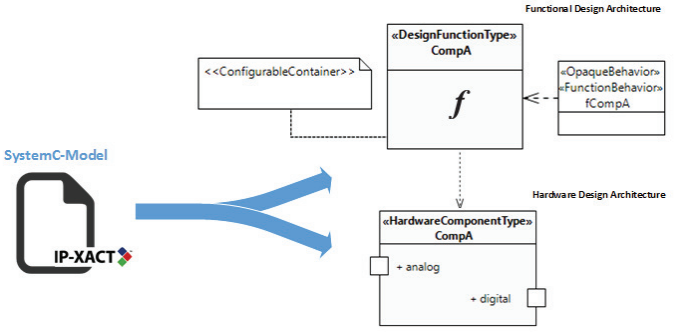
SystemC is defined by Accelera, a standards organization in the area of electronic design automation (EDA). It is an open standard modeling language and has been also approved by the IEEE standards association. SystemC is defined by several levels of abstraction. On the transaction level modeling level (TLM), a very high level simulation, the focus lies on communication and functionality. This serves as a golden reference for lower level hardware models (RTL). The RTL level included very detailed models where the components are connected through signals with pins. SystemC enables the design teams to have a fundamental understanding of the system at an early stage of the design process. Due to its high flexibility, it enables the representation of a complete system.

SystemC tries to bridge the gap between hardware description language (HDL) and object-oriented language (OOP). While SystemC is commonly used in the context of a system on chip or to model several components in a system, it is usually limited to the digital domain. In order to address complex systems with digital and analog parts, an extension was introduced called SystemC-AMS. This extension enables the simulation of continuous time, discrete time and discrete event behavior of analog/mixed-signals simultaneously. Nevertheless, SystemC lacks a visual representation for interacting with different stakeholders and their requirements. Because of its C++ background, SystemC is object-oriented and has a lot of similarities with UML and EAST-ADL, that supports the part,port and connector principle. This has also been acknowledged in publications such as[7] and makes the linking with EAST-ADL models intuitive. Because of this and its wide acceptance in the industry as well as availability, SystemC was chosen as the primary simulation language in our approach. Our methodology bridges the gap between model-driven design in EAST-ADL and executable models in SystemC.

The executable models used in our approach are models on different abstraction levels and in different versions. The generic models have the potential to be used in various domains and support reusability. The detailed models are refinements of the generic models and are to be used in special domains. Dependent on the domain and abstraction level, the models are built in SystemC(-TLM) and SystemC(-AMS).

5. System Libary

To avoid the design and simulation of larger systems from scratch and to achieve reusability, our methodology provides a SystemComponentLibrary (SCL). This library includes all major components for the simulation of systems in the automotive domain. Furthermore, it includes components in different versions and on different abstraction levels. A component in our library consists of two models, a structural model that describes the hardware structure and a functional model that describes the behavior. Both models have the same name but are tagged with a different type. The structural model is described by the EAST-ADL stereotype *HardwareComponentType* and owns the digital and analog ports of the hardware, tagged by *IOHardwarePin*. This mechanism also makes it possible to detach the digital components from the analog components. The behavior of the component is tagged by *DesignFunctionType*. This model owns the *FunctionBehavior*, which contains the kind and path to the behavioral description (SystemC model). In addition, the *DesignFunctionType* is tagged with *ConfigurableContainer*, which defines parameter and values of the component. These parameters may vary depending on the use case and verification methodology. To illustrate the togetherness, the functional model is allocated on the structural model with *FunctionAllocation*. Figure 1(a) depicts this approach. Since the creation of models for the SCL is a cumbersome task, a Text-to-Model converter helps to generate EAST-ADL models from SystemC code or even IP-XACT files. This converter generates the structural and behavioral models, with ports, parameters and allocation to the files. The mapping from SystemC/-AMS to EAST-ADL models is described in Table 1(b).



(a)

EAST-ADL		SystemC	
Stereotype/ Attribute	Type	Class/ Attribute	DataType
HWComponentType		sc_module	
IOHardwarePin			
type/direction	analog in analog out digital in digital out	sca_in sca_out sc_in sc_out	
HardwarePortConnector		sc_module	
busSpeed	float	txrate	double
busType		bustype	
DesignFunctionType		sc_module	
FunctionBehavior		class.dll	
ConfigurableContainer			
Type		type	
Attribute		attribute	

(b)

Fig. 1. (a) SystemComponentLibrary (SCL) text-to-model converter (b) mapping of EAST-ADL to SystemC models

To increase the reusability and provide good support for developers, the SystemComponentLibrary is built as an Eclipse plugin. New models can be generated and added to the library. Updates for components can be easily checked by updating the library from the server. This helps to support design teams by adding new components, and keeps the library consistent.

6. Methodology

Our methodology for the execution of EAST-ADL models is composed of four phases as depicted in Fig.2: Design-Phase, Build-Phase, Connect-Phase and Run-Phase.

6.1. Design Phase

The first part in our methodology is the system design. The designer creates an EAST-ADL design level class where the top-level is modeled. This class describes the overall architecture of the system. It is composed of the functional and hardware architecture model instances from the SystemComponentLibrary. These sub-systems are connected together by ports according to their specification. Due to the EAST-ADL port capability, these ports are checked in advanced by the framework in the correct direction of the dataflow or type (digital, analog). To trace the

dataflow, *Scope* models are added to the functional design and connected to functional ports. These *Scopes* collect the monitored data and stores them in trace files to compare the results from different tests and testbenches in a latter step. Our framework also provides a mechanism to encapsulate existing components and to raise the abstraction level of the whole system. Smaller systems that model the interior design of the class can be merged to a more simplified model. This helps the designer to have a better view of the system, without having too many details in the models on system level. This mechanism allows us to abstract the complexity of components. The whole eVehicle system, also referred to in our case as Design Under Test (DUT), is provided with several connectors. These signals required for testing and debugging are brought out to the ports of the top class itself. This has the advantage of connecting testbenches to the DUT for testing various scenarios of the electric car and also for monitoring performance. The outcome of the design step is a netlist that also serves as configuration and parametrization for the simulation. It is the starting point for the build phase.

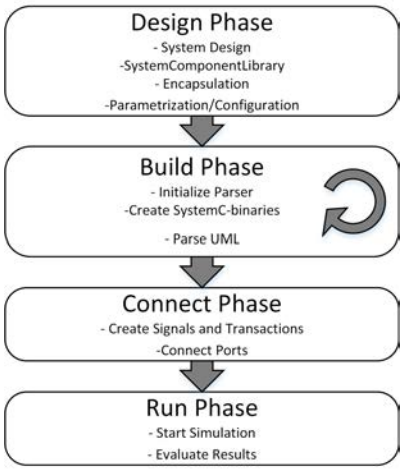


Fig. 2. Methodology for executable SystemC models from EAST-ADL design

6.2. Build Phase

The heart of our Build-Phase is the self-defined parser-methodology. The purpose of the parser is to translate a UML model defined in one or more files to a single SystemC system. This is done at run-time and does not require compilation of the resulting model. When the parser is initialized it requires the name of the top-class of the model in the diagram as well as the name of the UML file that contains the model. Starting from the root node of the UML model, each child of a node is parsed and returned. As single systems can be composed of more detailed sub-systems, we had to define a loop to find all properties and ports of each root note. Each node found in the UML file is treated as the new root node and each sub-system is created before moving on to the Connect-Phase. Each system may contain any number of sub-systems, therefore this step is done in several iterations till all properties of the root node are found. In order to keep the framework extensible a DLL-based plugin system is used. All the information is stored in a ConfigStore map.

6.3. Connect Phase

In this phase, the connector objects are created to link the different instances in the build phase. Depending on their nature, the connector objects can be signals or transactions. It is important to notice, however, that UML allows multiple 1:1 connections per port, SystemC merely allows a port to be bound once but a signal may connect any number of ports (basically 1:n as only one driver is allowed per signal). As a means of handling these issues, both ends of each connector are tagged using an ID. Instead of creating new signals for connecting to a used port, the old signal is reused.

6.4. Run Phase

After all nodes, ports and properties of the UML file have been found by the parser, and the SystemC instances have been created and connected, the simulation is started. The results of the monitored signals via *Scopes* are saved as trace files. These files contain all the relevant information required for the verification of the model or to evaluate the behavior of the model for different parameters and/or implementations for the system. Besides this, a logic can also be added to the system to react to certain events such as stopping the simulation in case of a signal, violating the given constraints, or the system running out of energy (for a battery powered system). An implemented dialogue is also used to configure the settings for the simulation such as duration or timestep (resolution).

Using the Toem Impulse plugin[12] for Eclipse, the results are presented in a graphical form. The results can be displayed in the desired manner, dependent on the nature of the simulation, e.g. analog interpolation for real values and numeric representation for digital signals in a hierarchy that allows for easy interpretations. The results may be verified against the known or expected behavior of the (physical) system modeled. If the system behaves as expected, it can be used for further analysis or verification (e.g as a golden reference model or synthesis-able).

7. Example Case Study: Electric Vehicle Simulation

We have applied our methodology to an industrial use case, an electric vehicle (eVehicle) system provided by CISC Semiconductor, to more fully illustrate its innovative capabilities and benefits. As more and more vehicles are now powered by Li-ion batteries, the challenge for engineers to ensure reliability and fault tolerance is also greatly increasing. It is crucial that the battery management systems (BMS) measure voltage, temperature and current of the battery very precisely to ensure safe operating conditions. This information must be forwarded to a system wide controller network to ensure a reliable and fully utilized system. Problems with overheating or even explosions have been frequent in the past. The main cause of these problems was an excessively high energy intake from regenerative braking or harsh environmental conditions. Management systems and mechanisms are thus essential to assure that persons are not put at risk and that no damage is caused. The overall system model of the eVehicle is depicted in Fig.3. It is composed of the *battery*, *controller*, *inverter*, *dc-motor*, *power train controller (PTC)* and the *battery management unit (BMU)*. The *driver* provides the desired speed for the eVehicle. This can be set according to standardized maneuvers such as the New European Drive Cycle (NEDC). The *controller* is a model for a PI state-space controller and maintains a constant speed based on the information about the state variables, motor armature current and motor-speed. The *inverter* model implements an inverter function for a PM-DC motor driving stage. It compares the actual battery voltage and the requested controller voltage to maintain the PM-DC motor terminal voltage. The *battery* model simulates the behavior of a Li-ion battery pack composed of a defined set of single cell Li-ion batteries. The appropriate number of single cells is connected in parallel and series to obtain the necessary capacity and terminal voltage. The battery pack's terminal voltage is calculated based on the defined parameter and the battery current. A BMU is connected to the battery to measure voltage, current and temperature of the cells/modules. The BMU computes the SOC, State-Of-Health (SOH) and is responsible for cell balancing, cell protection and demand management of the battery. These computed values are then processed via a CAN controller as digital values to the power train controller. In addition, the external load environmental conditions like temperature can be changed during the simulation.

The *Design Level* of the Design Under Test (DUT) contains the *Functional*, *Analog* and *Digital Design Architecture*. On the functional level, the components are tagged with *DesignFunctionType* which describes the behavior of the hardware. To provide stimuli and monitoring function, ports are brought outside of the DUT to connect various verification components like driver, monitor or scoreboard. Every *DesignFunctionType* on this functional level is allocated on its hardware counterpart on the hardware/digital architecture level as described in Section 5. The hardware components are tagged with stereotypes from the EAST profile such as *Node*, *Sensor*, *Actuator* and *ElectricComponent*. Depending on the nature of the port, the *IOHardwarePinKind* attribute is set to analog or digital. Each component is configured and parametrized through *ConfigurableContainer*.

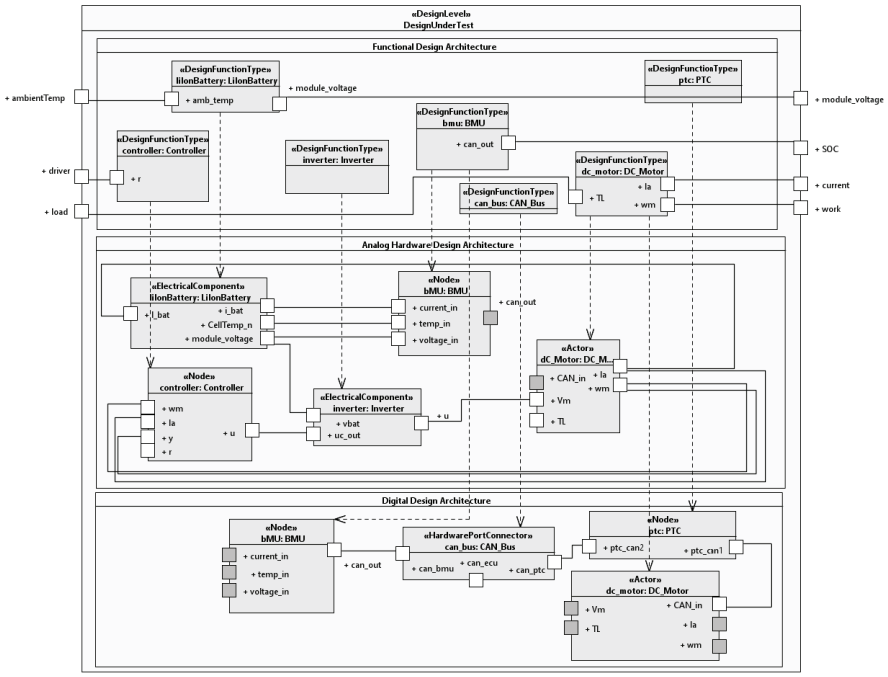


Fig. 3. EAST-ADL Design Level of the eVehicle simulation

8. Results

The results of our simulation-based verification within EAST-ADL are depicted in Fig.4. It shows the analog and digital signals which are monitored by Universal Verification Methodology (UVM) components. The DUT was stimulated by a driver with different driving scenarios. The exact simulation was also built as a Simulink model to compare our results with a golden reference model. The output of the Simulink run is referenced as *golden_ref* signal. Signal *deviation* shows the difference between signal *work* of both simulations engines. Only when it comes to a step in signal *load* there is a peak in the deviation of about 0.5 percent. This occurs because the SystemC simulation kernel requires an additional delay at this step, where Simulink, with its centralized timing solver, does not. This produces a short shift between both signals but ends up, after a timestep, with the same results. The average error between the SystemC and Simulink signal is 0.0081 percent. Both simulations have the same accuracy with a fixed timestep of 1×10^{-3}s.

9. Conclusions

In this paper, we presented a model-based simulation framework for verification of electric/electronic systems. We used the capabilities of EAST-ADL for a model-based design, to simulate analog and digital components in the automotive domain. With the help of our tool-aided methodology, we achieve simulation of systems seamlessly integrated into the design flow of ISO26262. Especially regarding functional safety, a model-based approach helps safety engineers to have an augmented view, to understand problems in the design and development of safety-critical systems. Through EAST-ADL models, the behavioral models in SystemC can be configured or even reconfigured for different testcases. A model library was introduced that helps to speed up the design and development process and raises reusability. With the help of UVM-like components, the whole system can be verified by methods like constraint random verification. Due to the Eclipse plugin mechanism, every Papyrus editor is now capable of executing their EAST-ADL models by installing our plugin. This tool called SHARC (Simulation and verification of HierARChical

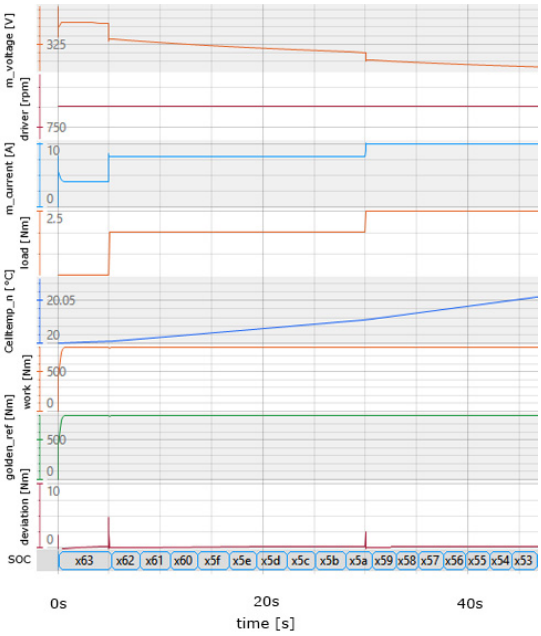


Fig. 4. Output-trace of the eVehicle simulation, the signals are compared to a golden reference

(embedded) systems) will be published for download and also used for educational purpose. To show the benefit of our framework, the tool-aided methodology was applied to an automotive use case of a battery management system. Another focus will be the resource handling of behavioral models for multi- and many-core applications. Because of its small memory footprint and fast execution time, this simulation environment will be used additionally for parameter variation in cloud-based environments.

Acknowledgments

The approach presented above is an experiment undertaken in the framework of OpenES CATRENE Project: CA703 - 2013 research program supported by the FFG (Austrian Research Promotion Agency), project-number 843380 in tight cooperation with CISC Semiconductor.

References

1. ETAS Embedded Systems Consulting: Electronic Control Unit (ECU) - Webinar Basics of Automotive ECU 2014;:1–30.
2. EAST-ADL Association. 2014. URL: `http://www.east-adl.info/`.
3. ISO 26262, . Road vehicles-functional safety-Part 5: Product development at the hardware level 2011;.
4. Maurer, M., Gerdes, J.C., Lenz, B., Winner, H.. Autonomes Fahren: Technische, rechtliche und gesellschaftliche Aspekte. Springer Open; Springer Berlin Heidelberg; 2015. ISBN 9783662458549.
5. Weissnegger, R., Kreiner, C., Pistauer, M., Römer, K., Steger, C.. A Novel Design Method for Automotive Safety-Critical Systems based on UML/MARTE. In: *Proceedings of the 2015 Forum on specification & Design Languages*. Barcelona, Spain; 2015:177–184.
6. Eclipse, . Papyrus. 2015. URL: `https://www.eclipse.org/papyrus/`.
7. Weiss, G., Zeller, M., Eilers, D., Knorr, R.. Approach for Iterative Validation of Automotive Embedded Systems. *Models 2010 ACES-MB Workshop Proceedings* 2010;:69–83.
8. Marinescu, R., Kaijser, H., Mikucionis, M., David, A., Seceleanu, C., Henrik, L.. Analyzing Idustrial Architectural Models by Simulation and Model-Checking. *Formal Techniques for Safety-Critical Systems* 2014;419:189–205. doi:10.1007/978-3-319-05416-2.
9. Taha, S., Radermacher, A., Gérard, S.. An Entirely Model-Based Framework for Hardware Design and Simulation. *International Federation for Information Processing (IFIP)* 2010;:31–42.
10. ATESST. 2010. URL: `http://www.atesst.org`.
11. maenad.eu. 2014. URL: `http://www.maenad.eu/`.
12. Toem Impulse. 2016. URL: `http://toem.de/index.php/projects/impulse`.